PAPER Special Section on Data Engineering and Information Management

Accordion: An Efficient Gear-Shifting for a Power-Proportional Distributed Data-Placement Method**

Hieu Hanh LE[†*a)], Satoshi HIKIDA[†], *Nonmembers*, *and* Haruo YOKOTA[†], *Member*

SUMMARY Power-aware distributed file systems for efficient Big Data processing are increasingly moving towards power-proportional designs. However, current data placement methods for such systems have not given careful consideration to the effect of gear-shifting during operations. If the system wants to shift to a higher gear, it must reallocate the updated datasets that were modified in a lower gear when a subset of the nodes was inactive, but without disrupting the servicing of requests from clients. Inefficient gear-shifting that requires a large amount of data reallocation greatly degrades the system performance. To address this challenge, this paper proposes a data placement method known as *Accordion*, which uses data replication to arrange the data layout comprehensively and provide efficient gear-shifting. Compared with current methods, Accordion reduces the amount of data transferred, which significantly shortens the period required to reallocate the updated data during gear-shifting then able to improve the performance of the systems. The effect of this reduction is larger with higher gears, so Accordion is suitable for smooth gear-shifting in multigear systems. Moreover, the times when the active nodes serve the requests are well distributed, so Accordion is capable of higher scalability than existing methods based on the I/O throughput performance. Accordion does not require any strict constraint on the number of nodes in the system therefore our proposed method is expected to work well in practical environments. Extensive empirical experiments using actual machines with an Accordion prototype based on the Hadoop Distributed File System demonstrated that our proposed method significantly reduced the period required to transfer updated data, i.e., by 66% compared with an existing method.

key words: *energy-aware, power-proportionality, data-placement, HDFS*

1. Introduction

In the era of Big Data, distributed file systems, such as Google File System [2] and Hadoop Distributed File System (HDFS) [3], have been widely used for efficient processing over a huge amount of unstructured data. In such systems, not only high performance but also power consumption have gained much attention from both academia and industry. Among this, power-aware distributed file systems are progressively moving towards power-proportional design [4]. The power-proportionality in these systems can be achieved using well designed data placement methods to control the total number of active nodes to store and retrieve the data [5]–[7]. The common idea in these methods is to carefully place the data's replicas at organized nodes. The system divides all of the nodes into a set of small separated groups. These groups are then configured to operate in multiple gears where each gear contains a different number of groups, and offers a different level of parallelism and aggregate IO [5]. A higher gear has a larger number of groups of active nodes. For example, if there is a 30-node system which is managed to operate in three gears. Gear 1 contains 10 active nodes, Gear 2 contains 20 active nodes and Gear 3 contains 30 active nodes. Accordingly, in a higher gear, the system consumes more power, although it can deliver higher performance with suitable processing. Leveraging this mechanism, the system can manage its power consumption and performance so it can deliver power proportionality.

However, the current power-proportional data placement methods do not well consider the effects of gear-shifting on the performance of the system. During operations, the system may have to update the datasets modified in a low gear when a subset of the nodes was powered off. When the system moves to a higher gear to gain a better power-proportionality by reactivating inactive nodes, it needs to provide the new power proportionality in serving the requests from clients. Therefore, in the background, the system has to internally re-transfer the updated data to the reactivated nodes to share the load equally among all of the active nodes in order the obtain better performance. Inefficient gear-shifting with the large amount of re-transferred data is believed to greatly degrade the performance of power-proportional distributed file systems.

To address this problem, in this paper, we propose an efficient and flexible power-proportional data placement method known as Accordion, which aims to improve the efficiency of gear-shifting by reducing the amount of re-transferred data. Accordion shares the node organization of other methods because it supports a multigear operational mode where each gear is combined from different numbers of different groups of nodes. However, the locations of the primary data in the dataset are different with Accordion. At first, the primary data in the dataset are located equally among all of the nodes in the system. Next, the data are replicated so each of its replicas is located on nodes that belong to different groups. As the primary data are located at all nodes, when the modified dataset is updated (or appended) in low gear, part of the primary data in the updated dataset is already stored on the active nodes. Hence, only the remainder of the updated dataset, which can be assumed to be written to the deactivated nodes, needs to be written

Manuscript received July 27, 2014.
Manuscript revised November 28, 2014.
Manuscript publicized January 21, 2015.

[†]The authors are with the Department of Computer Science, Tokyo Institute of Technology, Tokyo, 152–8552 Japan.

[*]Presently, with Yokohama Research Laboratory, Hitachi Ltd.

[**]This paper is an extended version based on our paper proposed in BigData 2013 [1].

a) E-mail: hanhlh@de.cs.titech.ac.jp

DOI: 10.1587/transinf.2014DAP0007

temporally. Consequently, only this temporal part of the updated dataset needs to be re-transferred when the system shifts to a higher gear. In other existing methods, the overall updated dataset needs to be reallocated during gear-shifting. As a result, Accordion is expected to reduce the cost of updating during gear-shifting because it reduces the period of data movement.

Furthermore, the efficiency of this reduction is increased when gear-shifting is performed in a high gear. In such situation, with a lower number of inactive nodes, the amount of data reallocated is reduced, then shortens the gear-shifting period further. It restricts the degradation of the system performance when Accordion-based systems shift to a higher gear. Consequently, Accordion is capable of providing smooth gear-shifting for multigear power-proportional distributed file systems. This feature of Accordion is well analyzed by numerical analysis and verified through our empirical experiments.

Moreover, Accordion also benefits from a load balancer that aims to share the workload among all of the active nodes. One of the main goals of the load balancer is to allow all of the active nodes to serve the same amount of data when responding to requests. However, another important factor that highly affects the performance of the file system is the timing distribution when active nodes serve requests. It should be well balanced in order to provide a higher I/O throughput performance. For example, if two nodes serve 20 GB of data, generally it would be more efficient if two nodes served 10 GB simultaneously rather than sequentially. The different locations of the primary data mean that Accordion can both balance the amount of data requested and the service timing on the active nodes. It is observed through our empirical experiments based on actual machines.

Additionally, it is easier to apply Accordion in a practical environment as the configuration, such as the number of active nodes in each gear, is more flexible. Although Rabbit [6] is the first method to achieve power proportionality in a HDFS, in addition to the inefficient gear-shifting due to the large amount of moving data, it also has a strict constraint where the number of nodes in each group is defined by an exponential function. In Accordion, this constraint is more flexible because there is only a simple constraint that the total number of nodes in the system should be an even number.

Last but not least, it is recognized that the advantage of Accordion is vital especially for commodity-off-the-shelf-based systems in which the power consumption has not been well optimized. In such systems, the power consumption of the systems generally has not gained enough support from modern technologies such as low energy CPU, memory or disk. For example, the power consumption of the investigated file server in idle time is still large compared with in active time (more than 75% as in [8]).

Our contributions are summarized as follows.

- We propose a distributed data placement method known as Accordion, which utilizes data replication to deliver efficient gear-shifting in power-proportional distributed file systems. The different approach of locating the primary data in Accordion reduces the amount of data updated during gear-shifting, hence shortens the time required to reallocate data.
- Accordion is applicable to smooth gear-shifting in multigear systems because it increases the performance efficiency when gear-shifting is performed in a higher gear.
- Accordion increases the I/O throughput performance because it improve the parallelism effectively distributes both the timing of serving requests and the amount of data requested.
- Accordion is expected to work well in practical environments because it does not require a strict constraint on the number of nodes in file systems.
- In order to evaluate the efficiency of Accordion, we performed extensive empirical experiments using a maximum of 24 nodes with an Accordion prototype on HDFS, which is widely used as the distributed file systems to support MapReduce [9] for efficient Big Data processing. The empirical experimental results showed that Accordion significantly improved the performance by 66% compared with Rabbit and Sierra [7], two of the most standard power-proportional data placement methods.

This paper is an extended version of [1] as it included the important parts of reporting the empirically experimental results which show the applicability of Accordion to provide the smooth gear-shifting in high-gear distributed file systems. Furthermore, the high efficiency of Accordion relating to the time to reallocate data is furthered compared with Sierra, which was newly implemented to HDFS.

The remainder of this paper is organized as follows. Related studies are discussed in Sect. 2 and the design of Accordion is described in Sect. 3. Section 4 presents a performance evaluation of our proposed method. Our conclusions and future work are discussed in Sect. 5.

2. Related Work

In this section, at first we reviewed some of the existing data placement methods in the field. Then, we described in more detail Rabbit and Sierra because they are used as comparative methods in the experiments.

Rabbit [6] was the first method to provide power proportionality to an HDFS by focusing on the read performance. Rabbit uses an equal work–data-layout policy based on data replication. The primary replicas are stored evenly among the primary nodes. The remaining replicas are stored on additional and increasingly large subsets of nodes. Each node in the subsets has a fixed order and it stores a number of blocks that is inversely related to its order, which guarantees that the system can distribute the workload equally among all of the active nodes. However, Rabbit still does not support the write workload so it cannot consider the cost of reflecting the updated data in a low gear.

Sierra [7] was designed as a power-proportional distributed storage system for general data centers where a replicated object store supports the write-and-read workloads during multigear operations. This method guarantees the write availability in a low gear by exploiting the concept of write off-loading [10], which was motivated originally by the aim of saving power by spinning down unnecessary disks. This method allows write requests on spun-down disks to be redirected to other active disks in the file system. Thus, this technique increases the spin-down duration, thereby providing additional power savings. This method may be considered as a solution for multigear file systems to deal with updated data in a low gear. Sierra can deal with write requests in a low gear, but it is still not optimized to reflect the updated data efficiently when the system moves to a higher gear.

In previous studies [11], we conducted a simple evaluation of Rabbit and PARAID [5] to identify an appropriate data placement approach to support efficient gear-shifting in power-proportional systems. PARAID uses a skewed pattern to replicate and stripe data blocks to the disks. This facilitates adaptation to the system load by varying the number of active disks in the system. PARAID focuses only on the RAID unit and it is unreliable when adapting to a distributed environment.

We also proposed an architecture known as NDCouplingHDFS [12], [13] to facilitate the efficient reflection of updated data in a power-proportional HDFS. NDCouplingHDFS focuses on coupled metadata management and data management on each HDFS node, which localizes the range of data maintained by the metadata in an efficient manner. This reduces the cost of managing the metadata generated during changes in the system configuration. However, the data placement method was not considered in this study.

Other studies, rather than power-proportional designs, have aimed to reduce the total power consumption based on a trade-off with performance in general storage system [14]–[17].

GRAID [14] is similar to PARAID because it is also a green storage architecture that aims to improve the energy efficiency and reliability of a RAID unit. In this study, the data-mirroring redundancy of RAID10 has been extended by incorporating a dedicated log disk, which stores all of the updates since the last mirror disk update. Using this log disk information, the system only needs to update the mirroring disks periodically so it can spin down all of the mirroring disks in a low-power mode for most of the time, which saves energy.

Kim et al. [17] proposed a fractional replication method to balance the power consumption and system performance. In this method, the data placement layout was inspired by PARAID where fractional replication and downshifting of the operational modes to a lower gear saved power using a probabilistic prediction model based on historical observations.

One of the first attempts to improve the energy efficiency of HDFS was performed by Leverich et al. [15] who showed that it was possible to recast the data layout and task distribution of HDFS to allow significant portions of a cluster to be powered down while still fully operational. They also confirmed that the energy could be conserved at the expense of performance so there was a trade-off between the two.

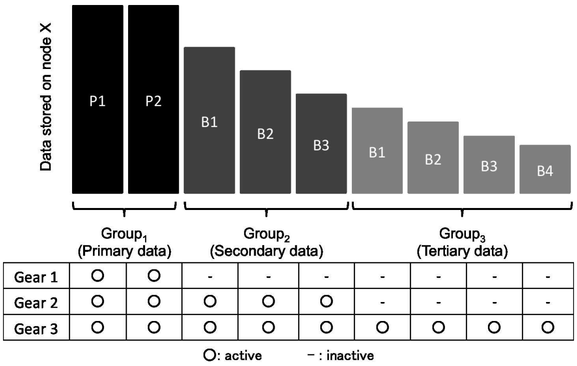
Later, Kaushik et al. [16] proposed an energy-conserving multiple zone approach for HDFS, which utilized life cycle information as data. They divided the storage clusters of the HDFS into hot and cold zones. The frequently accessed data were placed in the hot zone where all of the data nodes were active so they consumed power. The less frequently accessed data were placed in the cold zone, which allowed data nodes to be inactive. Thus, the power consumption was lower in the cold zone than in the hot zone. Some power savings were achieved in HDFS using this method. These techniques are able to be combined with our method.

2.1 Rabbit

Rabbit is a data placement for a power-proportional distributed file system that uses a data replication to control the power and performance of the system by achieving the the equal–work-policy in which the workload is evenly distributed among all active nodes.

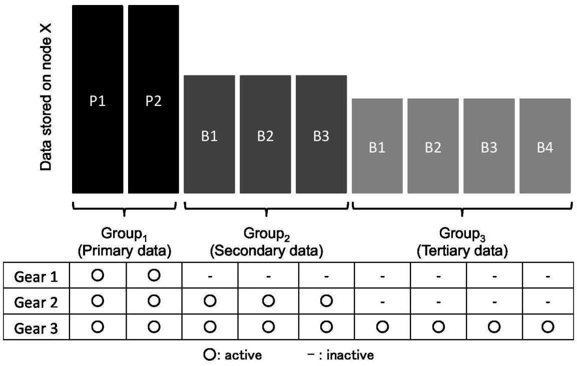
In Rabbit, the nodes in the system are divided into a number of separated groups and the replicas of the dataset are stored in a group unit. Assume that r replicas of B blocks of dataset D are to be stored in n nodes using G groups. Here, each node is numbering from 1 to n. Initially, a replica of all B blocks is evenly stored in the first primary p nodes in $Group_1$ (also called the primary group). Consequently, each node in $Group_1$ will contain $\frac{B}{p}$ blocks. The remaining $(r-1)$ replicas are distributed to the $(N-p)$ nodes such that a node $n_{(g,i)}$, where $g > 2$ and $p < i \leq N$, stores $\frac{B}{i}$ blocks. Here, using the constraint of keeping the number of replicas r small for a fixed number of nodes, Rabbit can guarantee that the number of blocks stored by the i-th node must not be less than $\frac{B}{n}$ for all $i \leq n$ when n nodes are active. Obeying this constraint makes it possible for the load to be shared equally among the active nodes. The performance of the system is therefore expected to be linear with the number of powered nodes, thereby fulfill the idea of equal–work-policy. However, this constraint also leads to the constraint of the number of nodes in groups from $Group_2$ to $Group_G$, as the number of nodes in each group is determined through an exponential function of r and p. The detail calculation can be found from [6].

Figure 1 depicts a heuristic illustration of a multigear system with Rabbit data placement. Here, the system has 3 groups of nodes and 3 gears. In Gear 1, only the nodes in $Group_1$ are active. The nodes in $Group_1$ and $Group_2$ are active in Gear 2. And all the nodes in $Group_1$, $Group_2$ and $Group_3$ are power on in Gear 3. The primary replicas of the



Gear 1	○	○	-	-	-	-	-	-	-
Gear 2	○	○	○	○	○	-	-	-	-
Gear 3	○	○	○	○	○	○	○	○	○

Fig. 1 An example of a three-gear system based on Rabbit data placement.



Gear 1	○	○	-	-	-	-	-	-	-
Gear 2	○	○	○	○	○	-	-	-	-
Gear 3	○	○	○	○	○	○	○	○	○

Fig. 2 An example of a three-gear system based on Sierra data placement.

dataset are stored on nodes on $Group_1$, the second replicas are maintained by nodes in $Group_2$ and the tertiary replicas are dedicated to nodes in $Group_3$. It should noted that the nodes in $Group_1$ stored the same amount of data while the other nodes contain a different amount of data.

2.2 Sierra

Like Rabbit, Sierra is also a data placement for a power-proportional distributed file system by leveraging the data replication. Sierra also organizes the replicas of the dataset such that, each replica of the dataset is stored in a group of nodes. The primary replicas are stored on a specific subset of nodes, called primary nodes ($Group_1$). Then, the secondary replicas are stored on a different group of nodes ($Group_2$) and so on. Sierra differs from Rabbit that each replica of the dataset is evenly distributed to all the nodes in each group. As a result, there is no constraint of the number of nodes in each group like in Rabbit.

Figure 2 shows an illustration of a multigear distributed file system based on the data placement in Sierra with 3 groups of nodes and 3 gears. It is recognized that all nodes in the same group store the same amount of data, which differs from Rabbit.

3. Efficient Gear-Shifting Method

In this section, at first, we describe the replication-based data placement method used by Accordion in detail. Next, we explain the load balancer for reads, which aims to distribute the workload among all of the active nodes and provide power proportionality. Finally, we present the processes that reflect updated data, which are written to the system in a low gear, when the system performs gear-shifting.

3.1 Accordion Design

Accordion was designed to provide power proportionality and a high data I/O throughput in cluster file systems that use commodity computer servers such as HDFS , Google File System. In Accordion, the files are divided into a large number of blocks and a number of replicas of each data block are distributed among the nodes of the cluster. The mapping of the file names to block identifiers and of the block to the block's locations are maintained by a separate metadata service.

Like other approaches, Accordion aims to control the power consumption of the system by dividing the nodes in a cluster into several separate groups. A system that uses the Accordion data placement layout can then operate in a multigear mode where each gear contains a different number of groups. The higher gears have more groups of nodes.

Power proportionality is achieved by Accordion's data placement policy, which is based on skewed replication, and a careful consideration of the effects of reflecting the modified dataset when the system moves from low to high gears. We describe the data placement policy in detail in the following parts and summarize the symbols used in Table 1.

3.1.1 Node Role Assignment

In a system that uses Accordion, changes in the operational modes of nodes are associated with changes in the shape of the bellows of an Accordion (musical instrument) after adjusting for effects on the transitions of a note or between multiple notes. When the system moves up to higher gears or down to lower gears, the active node ranges are expanded or reduced centrally.

In our method, the nodes are arranged geometrically in a horizontal array because the nodes that belong to lower groups are bounded by the nodes of higher groups. $Group_a$ is higher than $Group_b$ if $a > b$. For example, we assume that a system is operating in a two-gear mode with two groups of nodes, i.e., $Group_1$ and $Group_2$. The nodes from $Group_1$ are activated in Gear 1 and the nodes from $Group_1$ and $Group_2$ are turned on in Gear 2. In this case, $Group_2$ is higher than $Group_1$. Thus, the nodes of $Group_1$ are bounded by the nodes of $Group_2$.

Each node in the system is assigned its corresponding role in the group to perform data replication among the nodes in the system, i.e., to identify the destination nodes for

Table 1 Notations.

Symbol	Description
N	Number of nodes in the cluster
G	Number of groups in the cluster
$Group_g$	A group of nodes with index g ($g \in [1, G]$)
N_g	Number of nodes in group G_g
$Node_g^i$	A node with index i in group G_g ($i \in [1, N_g]$)
$Role(Node_g^i)$	The role of a node $Node_g^i$, is $Part_g^{order_i}$, where $Part$ is $Left$ or $Right$
$Left_g$	$Left_g = \forall Node_g^i \in Group_g, Role(Node_g^i) = Left_g{}^{order_i}$
$Right_g$	$Right_g = \forall Node_g^i \in Group_g, Role(Node_g^i) = Right_g^{order_i}$
B	Total number of blocks in a dataset
$V(B)$	Storage requirements to store a dataset with B blocks
$V(B)_g$	Storage requirements to store a dataset with B blocks in $Group_g$

Algorithm 1 Algorithm used to assign roles to nodes in the cluster.

```
Require: All the nodes of the system which are organized in $G$ groups ($Node_g^i$, $g \in [1, G]$, $i \in [1, N_g]$)
Ensure: $Role(Node_g^i)$
for all $Group_g$ from $Group_1$ to $Group_G$ do
  if $N_g$ is even then
    $middle = \frac{N_g}{2}$
    for all $i$ from $middle$ to 1 do
      $order = middle - i + 1$
      $Role(Node_g^i) = Left_g^{order}$
    end for
    for all $i$ from $middle + 1$ to $N_G$ do
      $order = i - middle$
      $Role(Node_g^i) = Right_g^{order}$
    end for
  else if $N_g$ is odd then
    $middle = \frac{N_g}{2} + 1$
    for all $i$ from $middle$ to 1 do
      $order = middle - i + 1$
      $Role(Node_g^i) = Left_g^{order}$
    end for
    for all $i$ from $middle$ to $N_G$ do
      $order = i - middle$
      $Role(Node_g^i) = Right_g^{order}$
    end for
  end if
end for
```

storing the backup data of a specific node. The role of each node is determined by the configuration in terms of the number of groups and the total number of nodes in each group. Algorithm 1 describes the function used to assign the roles to nodes:

$$Role(Node_g^i) = Part_g^{order}, \tag{1}$$

where $Part$ is either $Left$ or $Right$ and $order \in [1, N_g + 1]$.

Definition 1: For a $Group_{gA}$ that contains an odd number of nodes (N_{gA} is an odd number), there is a node that has two roles, i.e., $Left_{gA}^1$ and $Right_{gA}^1$.

An image of a multigear system using Accordion with organized nodes and roles are shown in Fig. 3. Gear 1 requires the nodes from $Group_1$ (a set of $Left_1$ and $Right_1$) to be activated while Gear 2 requires the nodes from both $Group_1$ and $Group_2$ to be activated.

3.1.2 Skewed Data Replication in Accordion

The three goals of Accordion are to provide power proportionality in the read performance, reduce the cost of reflecting updated data when the system changes gear, and guarantee the data reliability in all file system operating modes. Thus, the below policies are applied to satisfy these goals.

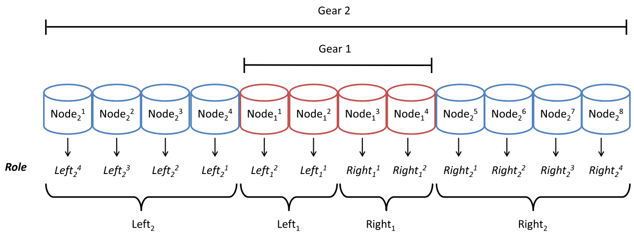


Fig. 3 Example of a two-gear system using Accordion where two groups of nodes have roles, where $G = 2$, $N_1 = 4$, $N_2 = 8$, and $N = 12$.

(1) Location of primary data

First, the primary data in the dataset are distributed to all of the nodes in the system. This means that each node stores the same amount of data.

(2) Location of backup data

Starting with the highest $Group_G$, the data stored in this group are replicated to the next lower group $Group_{G-1}$. Thus, the node with the role $Part_{G-1}^{order}$ in $Group_{G-1}$ is allocated to the backup data for the data from the nodes in $Group_G$, which have roles in the following range, if they exist.

$$[Part_G^{(middle-order)scale_G+1}, Part_G^{(middle-order+1)scale_G}],$$

where $Part$ is $Left$ or $Right$, $scale_G = \frac{N_G}{N_{G-1}}$, and

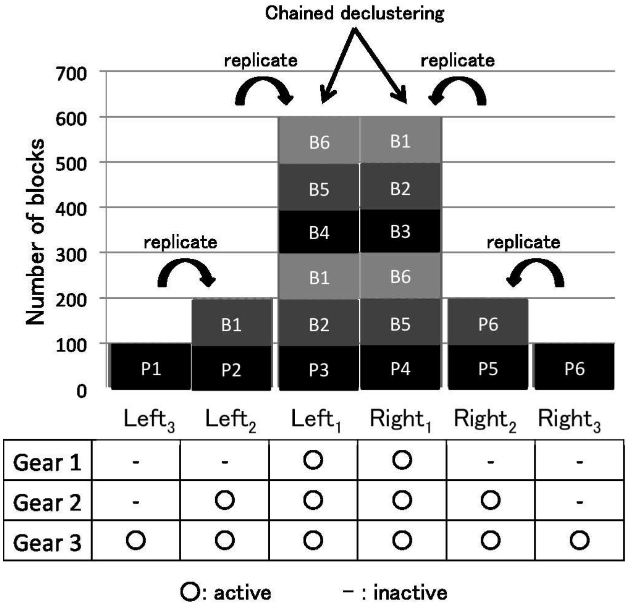
$$middle = \begin{cases} \frac{N_G}{2} & N_G \text{ is even} \\ \lfloor \frac{N_G}{2} \rfloor + 1 & N_G \text{ is odd} \end{cases} \tag{2}$$

For example, in Fig. 3, the data from the range [$Left_2^3$, $Left_2^4$] are replicated in the node with the role $Left_1^2$. This means that the data in $Node_2^1$ and $Node_2^2$ are replicated in $Node_1^1$.

The process is finished when the backup data for the nodes in $Group_2$ are replicated to the nodes in $Group_1$, the lowest group.

(3) Chained declustering at the smallest group

Chained declustering has been proved to provide superior



	Left$_3$	Left$_2$	Left$_1$	Right$_1$	Right$_2$	Right$_3$
Gear 1	-	-	O	O	-	-
Gear 2	-	O	O	O	O	-
Gear 3	O	O	O	O	O	O

Fig. 4 An example of a three-gear system based on Accordion data placement.

performance in the event of failure while maintaining a very high degree of data availability [18]. As a result, to guarantee the data reliability in the lowest gear, the chained declustering policy is applied to the smallest group ($Group_1$). Each node replicates its data to its neighbor node, which guarantees that all of the data in the dataset are replicated in the two neighbor nodes. In Fig. 3, the data (the primary data and the backup data) in $Node_1^1$ are replicated in $Node_1^2$, then from $Node_1^2$ to $Node_1^3$, and so on. In other power-proportional approaches, the data reliability in the lowest gear is often omitted because the data are not replicated.

Figure 4 shows an example of a three-gear system based on Accordion data placement described above where $B = 600$, $G = 3$, and $N = 60$ ($N_1 = N_2 = N_3 = 20$). $Group_i$ contains the nodes of $Left_i$ and $Right_i$. Gear 1 contains only the nodes in $Group_1$. To serve a request, Gear 2 requires the nodes in $Group_1$ and $Group_2$ to be activated, while Gear 3 requires all of the nodes in $Group_1$, $Group_2$, and $Group_3$ to be activated. The primary data in 600 blocks are stored equally among all of the nodes so each group locates 100 blocks initially. Next, the data in $Group_3$ are replicated in $Group_2$ and the data in $Group_2$ are replicated in $Group_1$. Finally, the chained declustering method is used to replicate the data in $Group_1$. Thus, the three groups contain 1200, 400, and 200 blocks.

3.1.3 Physical Data Placement on a Node Using Accordion

In Accordion, each node stores the primary data (the original data allocated) and the backup data (replicas of the data from other nodes) so the physical locations of these two types of data in the disks should be well designed. Normally, the data are stored physically in a number of sectors of the disks. The complexity of writing and replicating the data means that the orderless arrival timing of writing requests on each node promotes the discreteness of the physical locations of the sectors for the primary and backup data on the disks. However, on current disks, the seek time required to allocate the responsible sectors still degrades the I/O performance greatly, especially for discretely located sectors. Therefore, it is preferable to allocate the responsible sectors to sequential locations on the physical disks.

In this paper, we also propose an optimized version of Accordion known as *Accordion-with-Disk-Partition (Accordion-DP)* that uses a partitioning technique on each node, which logically locates the primary data and backup data in two separate parts of the disks. For example, in Fig. 4, in $Left_1$, P3, B1–B2 and B4–B6 are stored in separated partitions of the disks. Accordion-DP is expected to improve the I/O throughput performance of file systems because it reduces the seek time on the disks.

3.1.4 Storage Requirements

This section describes the total storage requirements for storing a dataset based on the skewed data replication policy used by Accordion. The total of blocks in a dataset is B and there are G groups, where each group contains N_g nodes. The total amount of data blocks $V(B)$ stored is calculated as:

$$V(B) = \sum_{g=1}^{G} V(B)_g, \tag{3}$$

where $V(B)_g$ is the number of data blocks located on the nodes of $Group_g$, $g \in [1, G]$. Initially, all of the blocks are stored equally among all of the nodes in the cluster so $V(B)_G = B \times \frac{N_G}{N}$, where $N = \sum_{g=1}^{G} N_g$ is the total number of nodes in the system. In addition to the original data, the nodes in $Group_{G-1}$ are also used to locate the replication data from the nodes in $Group_G$. Thus,

$$V(B)_{G-1} = B\frac{N_{G-1}}{N} + B\frac{N_G}{N} = B(\frac{N_{G-1}}{N} + \frac{N_G}{N}). \tag{4}$$

The amount of data in the remaining groups from $Group_{(G-2)}$ to $Group_1$ are similarly calculated. Finally, chained declustering is applied to $Group_1$ so the storage required by $Group_1$ is as follows:

$$V(B)_1 = 2\frac{B}{N}(\sum_{g=1}^{G} N_g) = 2B. \tag{5}$$

Substituting the values of $V(B)_g$ into (3) yields the following:

$$\begin{aligned} V(B) &= B(2 + \frac{N_G}{N} + (\frac{N_{G-1}}{N} + \frac{N_G}{N}) + \ldots + \sum_{i=2}^{G-i+1} \frac{N_i}{N}) \\ &= B\sum_{g=1}^{G} \frac{(g+1)N_g}{N}. \end{aligned} \tag{6}$$

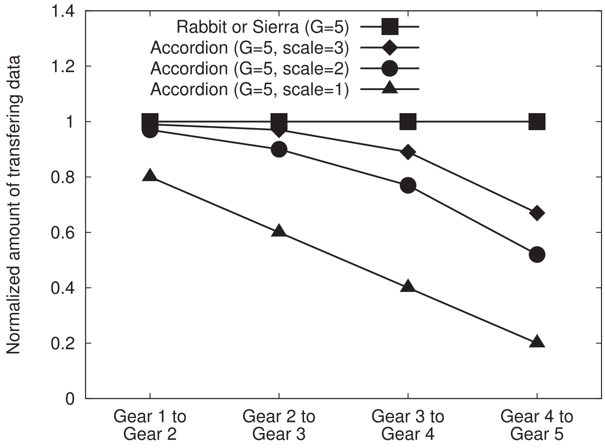


Fig. 5 The normalized amount of data transferred when gear-shifting is performed in each gear.

3.1.5 The Amount of Data Transferred During Gear-Shifting

In this section, we calculate and compare the amount of data that is transferred internally when the system changes its configuration using Accordion and other methods such as Rabbit and Sierra. In such methods, as each group contains one replica of the dataset, the entire updated dataset is transferred during each gear-shifting. In Accordion, however, shifting between each gear in the system only leads to variation in the subpart of the dataset located on the nodes that are reactivated during transitions.

From the calculations in Sect. 3.1.4, the amount of data moved when the system changes from Gear g to Gear $(g+1)$ is,

$$V_g(D) = D \sum_{i=g+1}^{G} \frac{N_i}{N}, \tag{7}$$

where G is the number of groups, N_i is the number of active nodes in $Group_i$, N is the total number of nodes, and D is the number of blocks updated in the dataset in Gear g. A constraint on Accordion is that the preferred number of nodes in $Group_g$ is a multiple of the number of nodes in $Group_{g-1}$. For a clearer understanding, we consider a simple case where $\forall g \in [1, G]$, $\frac{N_{g+1}}{N_g} = scale$ $(scale \geq 1)$. Substituting into (7), we obtain the following:

$$V_g(D) = D \frac{\sum_{i=g+1}^{G} scale^i}{\sum_{i=0}^{G-1} scale^i}. \tag{8}$$

Figure 5 shows the amount of data moved, which was normalized against the amount of the dataset that was updated during each gear shift with Accordion and other methods (Rabbit and Sierra), where the systems were configured to operate using five gears. The amount moved was always smaller in Accordion than Rabbit or Sierra and it became smaller when the system performed gear-shifting in a higher gear. The degradation of the system performance was reduced so Accordion may be applicable to smooth gear-shifting in multigear power-proportional file systems where the number of gears is high.

3.1.6 Skewed Dataset Distribution in Accordion

The skewed data replication in Accordion leads to an imbalance in the amount of data stored on each node. Like in Rabbit and Sierra, some nodes (in the smaller groups) store considerably more data than others (in the higher groups). However, this imbalance does not lead to considerable problem to the overall I/O performance as current hard disk drives normally make use only part of the capacity other than full of the capacity to provide the I/O requests. Given that the amount of unused storage is increasing [19] and that the energy problem is a greater focus, the provision of smooth gear-shifting is more important when delivering power-proportional systems.

3.1.7 Fault Tolerance

A specific algorithm to deal with node failures in a system is beyond the scope of this study. When a node fails, however, all of the nodes in the system are reactivated and the data from a failure node can be reconstructed based on the backup data. In a future study, we plan to provide a specific solution to this problem in more detail.

3.2 Load Balancer

When a request for a block is received from a client, the file system has to select the node that will serve the request because the block is replicated in multiple nodes in the system. To provide power proportionality when serving a dataset with multiple blocks, it is preferable to balance the load among all of the available nodes. The load balancing mechanism used in Accordion is basically the same as that employed by Rabbit. If n nodes are active when serving a B-block dataset, the goal of the load balancer is to make each node serve $\frac{B}{n}$ blocks. Similar to Rabbit, we define an ideal value for each node where $Node_g^i$ equals $\frac{B}{n \times contain_g^i}$. When the dataset is being read, the current hit of the node $Node_g^i$ is $\frac{served_g^i}{1 \times contain_g^i}$, where $contain_g^i$ is the number of blocks stored locally on $Node_g^i$, and $served_g^i$ is the number of blocks already served by $Node_g^i$. From the possible nodes that may store a requested block, the load balancer greedily selects the node where the distance between the current hit and the ideal value is the largest.

Although this mechanism is similar to Rabbit, the different policy of locating the primary data makes Accordion preferable to Rabbit. In Rabbit, each of the replica set of the whole dataset is stored separately in each group's node while in Accordion, there is always part of the dataset that is stored only in the lowest group's nodes. Consequently, the lowest group's nodes can serve the request from the beginning and hence it leads to less skew in distributing the load among nodes in Accordion than in Rabbit. It is explained in more detail and clearer in Sect. 4.4.2 through utilizing an empirical experiment.

3.3 Writing Data Using Write Off-Loading

When a subset of a group of nodes is deactivated in a low gear, the system can accept write requests from clients using the write off-loading technique. Next, when the system moves to a higher gear by reactivating nodes, it transfers the updated data internally in the background without stopping the processing of read serves from clients.

3.3.1 Writing New Data Using the Write Off-Loading Technique

When the system is in a low gear and it has to deal with requests to write new data to a previously stored dataset, the placement of data is performed according to its original policy. Because the system operates in low gear, certain parts of the new data cannot be written to their corresponding deactivated nodes. Hence, the system selects another node randomly from the active nodes to serve this request. Information about the data, the temporary node, and the intended node are saved in a log file. An example of the write off-loading process is shown in Fig. 6. In this example, $Node_2^2$ should serve the write request for data E according to the original data placement policy. However, it is powered off in a low gear so the system decides to use an alternative node, i.e., $Node_1^1$ is selected in this case.

3.3.2 Reflecting Updated Data

When the system changes to a high gear to serve a request for high-performance processing with the newly updated dataset, it needs to perform two functions. The first function is to transfer the data written in temporary nodes to their actual intended nodes. This can be achieved by reading the information in the log files. In Fig. 6, $Node_1^1$ is set to transfer data E to $Node_2^2$. The second function is to serve a request by scanning the new dataset in the storage system. When there is no need to correct the data layout, compared with a normal service in a high gear, the cost of allowing the system to operate in multiple modes to save power will depend greatly on the quantity of data that needs to be transferred.

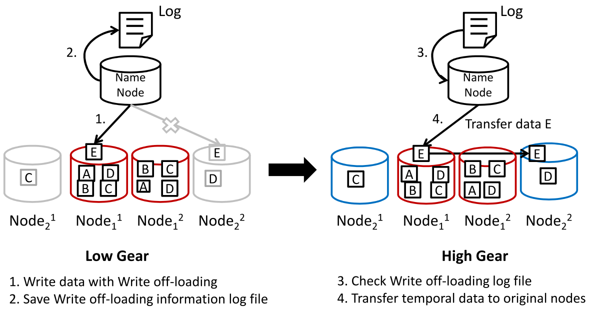


Fig. 6 Updated data reflection using write off-loading.

4. Experiments

We conducted an empirical experiment using actual machines to verify the methods proposed in this study. We chose Rabbit and Sierra, two of the de facto power-proportional data placement methods for distributed file systems as the comparative methods. First, we evaluated the efficiency of Accordion for smooth gear-shifting in multi-gear systems. Next, we compared Accordion with Rabbit and Sierra with respect to power proportionality during the performance of reads. Finally, we evaluated the power proportionality of Accordion using several configurations.

4.1 Implementation

In this study, we implemented a prototype of the power-proportional distributed file system Accordion based on a modified HDFS. For comparison evaluation, we implemented the data placements in Rabbit and Sierra from the scratch based on their published papers. We changed the current class used to select block locations via an interface where different data-layout policies can be performed. We also added the load balancing mechanism in to HDFS. The write off-loading policy was implemented in the low-power mode, which selects the temporal nodes for writing the new data for the currently turned off nodes. To guarantee the data reliability, block replicas were written to distinct nodes in all of the operation modes.

4.2 Framework Used for the Experiments

Our test-bed for the experiments comprised dozens of commodity nodes, which was based on the HDFS architecture with one NameNode, and a cluster of nodes for storing data. We were focused on the energy-aware commodity system so we used low power consumption ASUS Eeebox EB1007 machines, the specifications for which are provided in Table 2. In the experiments, The power consumption of the cluster of data storage nodes was measured using an AC/DC Power HiTESTER 3334 [20]. The interconnect was a 1000 Mbps switching hub and all of the inactive nodes were hibernating.

4.3 The Smoothness of Accordion in Multigear Systems

In this part, we verified the efficiency of Accordion relating

Table 2 Specification of a node.

CPU	TM8600 1.0 GHz
Memory	DRAM 4 GB
NIC	1000 Mb/s
OS	Linux 3.0 64-bit
Java	JDK-1.7.0

Table 3 Experimental environment.

#gears	6
#active nodes from Gear 1 to Gear 6	4, 8, 12, 16, 20 and 24
#files	420
File size [MB]	64
HDFS version	0.20.2
Block size [MB]	16, 32 and 64

to reflecting the updated data in multigear systems in which the number of gears is high. As discussed by the numerical analysis in Sect. 3.1.5, Accordion is believed to be applicable to perform smoother gear-shiftings than other existing methods as the amount of reflected data is thoroughly considered. This result is verified by the empirical experiments performed in this section.

4.3.1 Experimental Environments and Method

We compared Accordion with Sierra in terms of their execution times needed to reflect updated data when the system moved from a low gear to a higher gear. The reason we choose Sierra as a comparative method is that Sierra and Accordion shares a common feature of flexible configuration of the multigear systems, especially when the number of gear is high. In contrast, Rabbit requires very large numbers of nodes in higher gears because the number of nodes in each gear is determined through an exponential function.

The systems used in this experiment were operated using a six-gear configuration based on six groups of nodes. The numbers of active nodes from Gear 1 to Gear 6 were set to 4, 8, 12, 16, 20 and 24 consequently. The data set contained 420 files in which the file's size was fixed to 64MB. The block size used in HDFS was varied to 16 MB, 32 MB and 64 MB. Here, it is noted that, when the system achieves gear-shifting, the whole dataset is updated. As the number of files are 420, the size for the whole dataset is 26880 MB. We also perform another experiment reported in Sect. 4.5 to evaluate Accordion with varying the size of the updated data. Table 3 summarizes the above experiment settings.

Using write off-loading described in Sect. 3.3, the dataset is written when the systems operated in Gear 2. Specifically, the replicas of files that should be written to deactivated nodes would be temporally sent to other activating nodes. Here, from the fault tolerance point of view, we guaranteed the constraint that the replicas of the same file are not written to the same node. Next, we consequently shifted the systems to higher gears, step by step from Gear 3 to Gear 6. During each shift, the temporal data were retransferred from temporal nodes to original nodes that were reactivated according to each data placement method, Sierra or Accordion. The average execution times for reflecting the temporal data at each shift were measured and reported.

4.3.2 Experimental Results

Figure 7 describes the average execution time needed to move the updated data at several configurations of both Sierra and Accordion. From this figure, at each block size setting, it is seen that the measured execution times in Sierra were almost unchanged in all configurations. However, it could be also derived that Accordion outperformed Sierra in all configurations. Furthermore, the effect of Accordion became larger in the configurations in which the gear-shifting was performed at higher gears as the execution time becomes shorter. Accordion gained highest result in configuration Gear 5 to Gear 6 where it improved the execution time by 91% compared to Sierra when the block size was 16 MB.

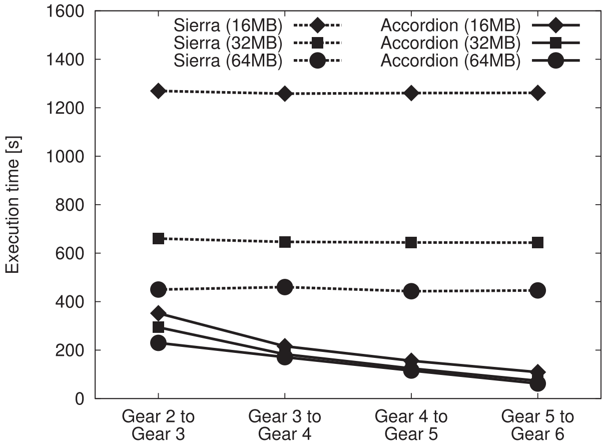


Fig. 7 Execution time for reflecting updated data.

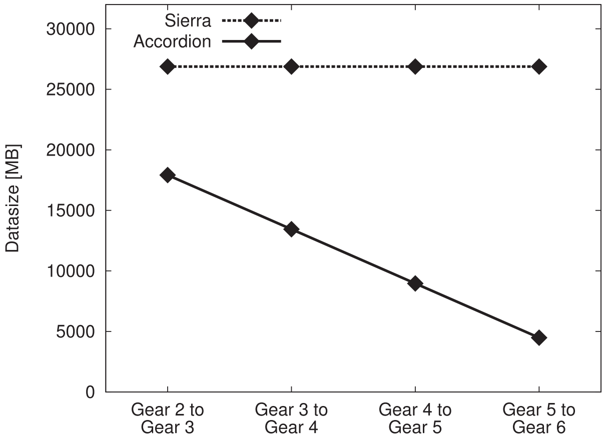


Fig. 8 Size of transferred data.

The reason of the above results is shown in Fig. 8 which presents the amount of reflected data at each configuration. In Sierra, the amount of reflected data are the same and equal the size of the dataset. When the system in Sierra shift to a higher gear through reactivating a group of nodes, it has to reflect all the updated data of that group. Due to the fact that each group contain one replica of the dataset, the amount of the updated data equals to the dataset's size. In contrast, the experimental results reported in this figure verified the numerical analysis that in Accordion, the size of reflect data always smaller than Sierra and becomes smaller when the gear-shifting is perform at higher gears. As a result, Accor-

Table 4 Experimental environment.

#gears	3
#active nodes (Accordion, Accordion-DP, Sierra)	2, 8, 20
#active nodes (Rabbit)	2, 7, 21
#files	420
File size [MB]	64
Block size [MB]	32

dion is believed to be applicable to smooth gear-shifting in multiple power-proportional file systems where the number of gears is high.

By changing the value of the block size, we could evaluate the effect of the number of blocks to the performance of Accordion during gear-shifting. Figure 7 also describes that in both Sierra and Accordion, the execution times were shorter with larger block size. The reason is mainly owing to the metadata management cost in HDFS. Larger block size, which leads to smaller the number of blocks in HDFS, decreased the cost of the metadata management in our experiments.

4.4 Power Proportionality of Accordion, Rabbit and Sierra

We evaluated the effectiveness of Accordion by determining the ratio of the throughput when reading a dataset and the power consumption of the cluster of data-storing nodes. We decided to elect two of the most state-of-the-art power-proportional data placement methods, Rabbit and Sierra, as two comparative methods.

4.4.1 Experimental Environments

We compared Accordion and Accordion-DP with Rabbit and Sierra in terms of their power proportionality when performing reads. These systems were operated using a three-gear configuration based on three groups of nodes. The constraint on the number of nodes in the gear configuration meant that the three gears in Rabbit contained 2, 7, and 21 nodes. To ensure a fair evaluation, the three gears in Sierra, Accordion and Accordion-DP had 2, 8, and 20 nodes. In this experiment, as in Rabbit, the read performance of scanning all the dataset is focused, then all 420 files of the dataset are read and each file is read once. The workload is distributed uniformly to all of the 21 clients as each client sequentially sends the requests of reading 20 separated files to the file systems. Here, the size of each file is 64 [MB]. The other information can be referred from Table 4.

4.4.2 Experimental Results

Figure 9 shows the throughput per watt to compare the read performance using the aforementioned dataset with three power-setting modes: Gear 1, Gear 2, and Gear 3. The results were the averages of five runs and the Linux buffer cache was cleared between runs.

There were not much differences in the results in Gear 1 and Gear 2 of the three configurations Rabbit, Sierra and Accordion. However, in Gear 3, it is seen that Accordion delivered approximately 10% better results than Rabbit because it gained better throughput as can be seen in Fig. 10 which shows the average throughputs for the configurations. The reason was that the better load balancer in Accordion-based increased the throughput while consuming less power (20 nodes vs 21 nodes in Rabbit).

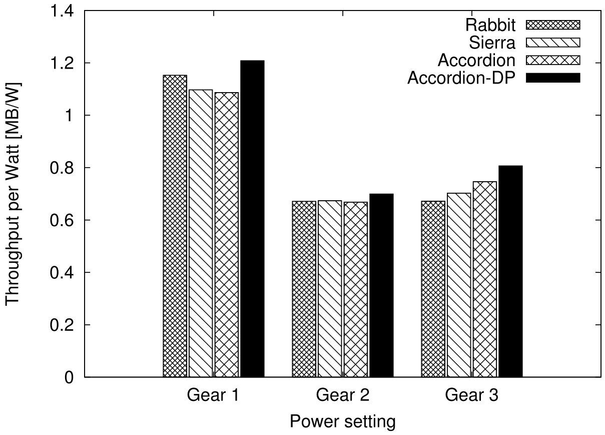


Fig. 9 Throughput per watt using Accordion, Accordion-DP, Rabbit and Sierra with various power settings.

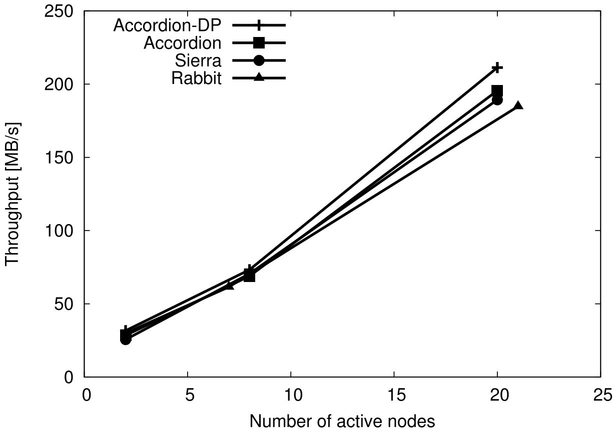


Fig. 10 Read performance using Accordion, Accordion-DP, Rabbit, and Sierra with various power settings.

The load balancer was the same in all configurations so the differences is explained by the timing distribution when each node served the requested blocks. Figure 11 shows the timing distribution of serving blocks at active nodes in the system for Rabbit and Accordion in Gear 2 and Gear 3. The blank part of the graphs at a node shows that the node does not serve any block. The results of Sierra were similar to the results of Rabbit. In each graph, the vertical axis is the NodeID while the horizontal axis shows the BlockID of the dataset. In Rabbit, the nodes in ranges $[1, 2]$, $[3, 7]$ and $[8, 21]$ consequently belong to $Group_1$, $Group_2$ and $Group_3$. In Accordion, the corresponding ranges are $[1, 2]$, $[3, 8]$ and $[9, 20]$. In Gear 2 of Rabbit, as there are only 7 active nodes from NodeID 1 to NodeID 7, there were no result of NodeID 8. The other blank parts of the graphs

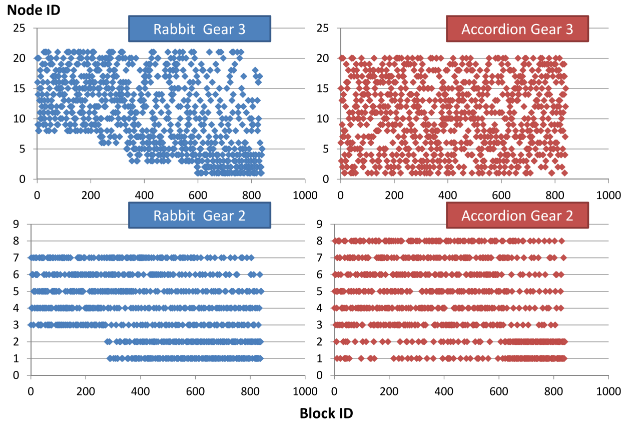


Fig. 11 Distribution of timing on the nodes when serving the blocks in Gear 2 and Gear 3 using Rabbit and Accordion.

of Rabbit at some nodes show that those nodes do not serve any block. The reason of this result lays in the data placement in Rabbit because both Rabbit and Accordion used the same load balancer.

In Rabbit and Accordion, the load balancer as described in Sect. 3.2, to guarantee that all the active nodes serve the same blocks from the dataset, determines the ideal value for each node. This ideal value for each nodes depends on the total number of blocks of the serving dataset and the number of the blocks stored at this node. Because the total number of blocks of the dataset is fixed, the decision of choosing which node will serve the block from the nodes that storing the replicas of the requested block, is depended on the number of blocks storing at those nodes. As a result, the common load balancer utilized in Rabbit and Accordion generally chooses the nodes of higher gear groups, which store less blocks than the nodes of smaller gear groups.

For example, in Rabbit the replicas of BlockID 100 are stored at NodeID 1 of $Group_1$, NodeID 3 of $Group_2$ and NodeID 10 of $Group_3$. Here, $Group_3$ is the highest gear group and $Group_2$ is higher gear group than $Group_1$. As a result, when the system operates in Gear 3, NodeID 10 is chosen; in Gear 2, NodeID 3 is chosen and in Gear 1, NodeID 1 is chosen to serve the BlockID 100.

Although Rabbit and Accordion utilize the common load balancer described above, it is the data placement that generated a different result in Fig. 10. In Rabbit, because replicas of the dataset are stored in group unit, it is ensured that a replica of every block is stored in a node of every group, as in the above example. However, it is different in Accordion as there exists a number of blocks that theirs replicas are not stored at every group of the system. In the example described in Fig. 4, the replicas of blocks in P3 and P4 are only stored in nodes of $Group_1$ and the replicas of blocks in P2 and P5 are not stored in nodes of $Group_3$.

As a result, in Rabbit, the nodes in higher groups more aggressively serve the requests than the nodes in lower group at the start of the experiment (with smaller BlockIDs), and the nodes in lower group have to wait until the late of the experiment (with higher BlockIDs) to join the work. Whilst in Accordion, because there are blocks whose replicas only stored at nodes of lower groups, such nodes are forced to serve those blocks whenever they are called.

In this experiment, the benefit of the partitioning technique for separating the primary and backup data areas on nodes was confirmed as Accordion-DP yielded approximately 10% better results than Accordion (Gear 3). From the latter experiments, Accordion-DP was used to evaluate the effectiveness of Accordion.

4.5 Effect of Reflecting the Updated Data on the Performance

The goal of the experiment in this section is to evaluate the effect of the proposal in the event that when the system shifts from a low gear to a high gear, it serves the workload of read request of reading all the dataset while performing reflection of updated data in the background. Here, the workload of read request was the same as the workload used in Sect. 4.4 because we wanted to verify how the read performance is affected by the updated data reflection. It is achieved by comparing with the results of Rabbit, Sierra, Accordion-DP at Gear 3 reported in Fig. 10 in Sect. 4.4 in which no data reflection is occurred.

4.5.1 Experimental Method

The workloads are generated that they are closed to the actual operation of the multiple-gear distributed file systems, like HDFS. It is assumed that initially the file system was operated in a High gear and stored an initial dataset. Then, the system shifts to a Low gear for a specified power-proportional service agreements. During this period, this dataset was updated as new files are appended from the clients. The systems have to apply the write-offloading technique to serve this update. Next, the systems are shifted to the High gear in order to satisfy the higher throughput performance on reading the whole dataset from the clients. Here, the systems have to serve the reading request from the client while performing updated data reflection in the background. In this assumption, as we focused on the applications on the distributed file systems like HDFS, we choose the method to update the dataset as appending new files to the dataset and to read the dataset as scanning all the files in the dataset, which are considered very closed to the actual deployments of HDFS.

In order to realize the above descriptions, three kinds of workloads are used in this experiment. The first workload is for the data storing process, in which an initial dataset was written to the file systems when it operates in Gear 3. The second workload is for the data update process, in which an updated dataset was written to the file system when it operates in Gear 2. The third workload is for the data reading process, in which a reading dataset, which contains the initial dataset and the updated dataset, are read from the clients when the file system operates in Gear 3.

Table 5 Number of files in each dataset used in configurations.

Configuration	Small	Medium	Large
Updated dataset	70	140	210
Initial dataset	350	280	210

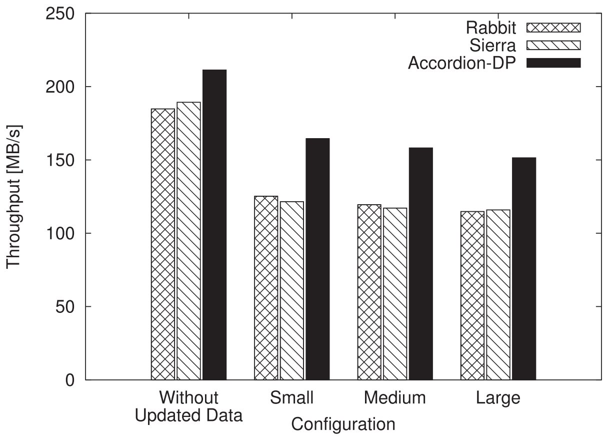


Fig. 12 Average throughput for scanning the dataset.

Because the performance degradation of reading throughput over the whole dataset is evaluated, the size of the reading dataset in the third workload is fixed to the same as in the workload used in the previous experiment (Section 4.4). As a result, the reading dataset contain 420 files, each file is 64 [MB]. Furthermore, as the effect of the updated data reflection, which is performed in the background when the systems serve the read request from the clients is evaluated, the sizes of the initial dataset and the updated dataset in three configurations are varied. In Small, Medium, Large configuration, the numbers of files of the initial dataset and the updated dataset are consequently set as (350, 70), (280, 140) and (210, 210) (Table 5).

The third workload for the data reading process is generated similarly as in the previous experiment (Section 4.4), i.e. uniformly distributed to clients. Here, the number of clients is 21, which equals the maximum number of active nodes in Rabbit for a fair evaluation. Each client queried separated and unique 20 files and the order of the files is randomly created.

4.5.2 Experimental Results

Figure 12 shows the experimental results as the average throughput while scanning the dataset in three cases: small, medium, and large amounts of transferred data. The results were compared with the results when the system performed the read requests in Gear 3 (without updated data), which were the same as the previous experiment. It is seen that configurations based on all of Accordion-DP, Rabbit and Sierra were affected significantly by the updated data reflection. The throughputs were degraded significantly by more than 20% (approximately 20% for Accordion-DP and 30% for Rabbit and Sierra). It became worse with higher amounts of re-transferred data. This confirmed the need to design the data placement method carefully to ensure efficient gear-shifting in power-proportional file systems.

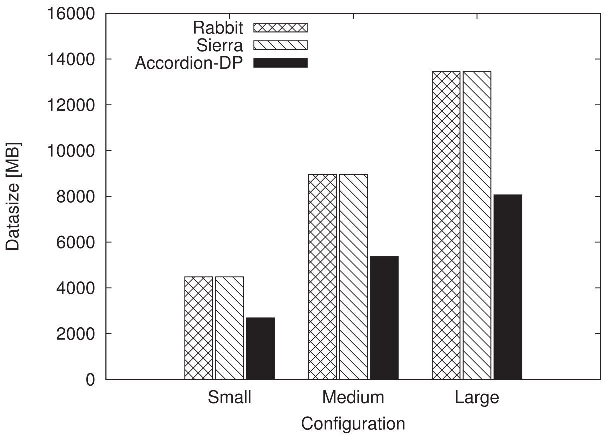


Fig. 13 The amount of data that had to be re-transferred when the system moved from Gear 2 to Gear 3.

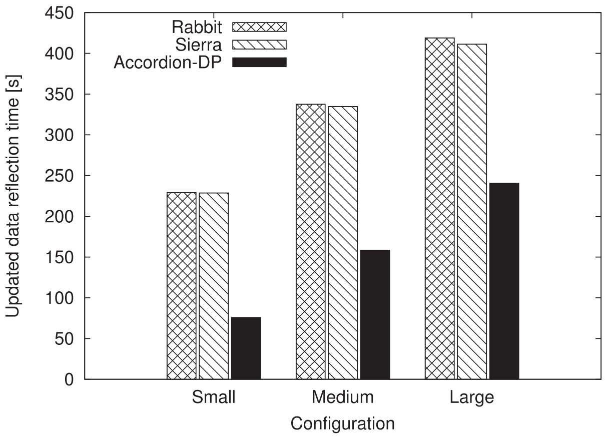


Fig. 14 The execution time for the data-re-transfer process when the system moved from Gear 2 to Gear 3.

The results also indicated that Accordion-DP improved the performance by 30% compared with Rabbit. The reason is with Accordion-DP, the amount of data re-transfer required was less than that with Rabbit and Sierra. With Accordion-DP, only part of the dataset written when the system operated in Gear 2 had to be re-transferred when the system moved to Gear 3. By contrast, the whole dataset had to be re-transferred with Rabbit as the placement policy located replicas of the dataset in each group.

Figures 13 and 14 show the amounts of re-transferred data and the execution times required to finish the re-transfer process. Figure 13 shows that with Rabbit and Sierra, the amounts of data in the small, medium, and large cases were 4480 MB, 8960 MB, and 13440 MB; while with Accordion-DP were 2688 MB, 5376 MB, and 8064 MB, respectively. As shown in Fig. 14, Accordion-DP significantly reduced the time required to finish the updated data-re-transfer process by up to 66% (Small case) compared with Rabbit and Sierra.

Table 6 Experimental environment.

#Gears	3
#active nodes Accordion-DP (4, 8, 12)	4, 8, 12
#active nodes Accordion-DP (4, 12, 20)	4, 12, 20
#active nodes Accordion-DP (8, 16, 24)	8, 16, 24
Number of files	420
File size	64 MB

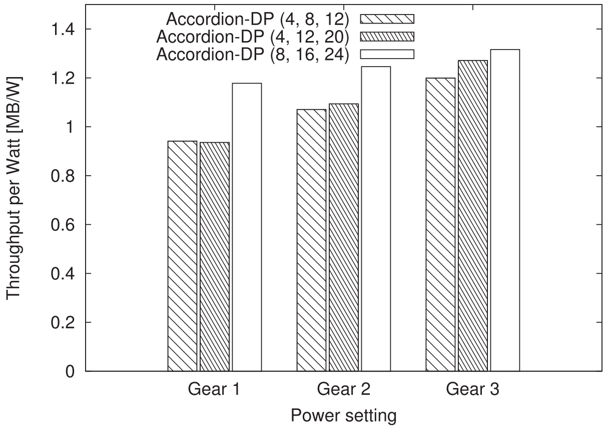


Fig. 15 The throughput per watt with Accordion-DP in several configurations.

4.6 Effects of Different Configurations of Accordion-DP

We also performed an experiment to evaluate the effects of different configurations of Accordion-DP on the power proportionality of the system. We used three configurations with different numbers of active nodes. The system was assumed to operate in a three-gear mode: Gear 1, Gear 2, and Gear 3. In the configuration Accordion-DP (x, y, z), x, y, and z indicate the number of active nodes in Gear 1, Gear 2, and Gear 3, respectively. The other environments were the same as those used in the previous experiment in Table 6.

Figure 15 shows the performance (throughput per watt) for the three configurations we evaluated. It is seen that larger configuration gained better throughput per Watt results at all power settings. The considerable reason is that the load at each node in larger configuration is lighter as the number of serving data becomes smaller. As a result, it can be suggested that Accordion is able to perform well with the high number of active nodes. In a future work, we want to verify this assumption by further experiments using a more number of nodes.

4.7 Discussion

The data placement strategy used by Accordion-DP ensures high power-proportionality performance of systems, especially when systems operate in a high gear mode with a larger amount of active nodes. It was well modeled through numerical analysis and verified through the empirical experiments. In Accordion-DP, the amount of data requested and the access timing of the active nodes are well distributed among all the active nodes so the I/O throughput performance scales well with the system's size. Furthermore, Accordion-DP was highly effective at reducing the cost of gear-shifting because it improved the performance by 66% when transferring the updated data.

Benefit from the smaller amount of the updated data, Accordion is believed to perform well for the write-intensive applications, in which the larger amount of re-transferred data will occur because of write-offloading. Moreover, although large amount of transferred data will occur, regarding to the primary data is focused, the write throughput for the applications when the system operates in a lower gear is improved compare with Rabbit and Sierra. As the primary data are also managed by higher gear groups instead of only the lowest gear group as in Rabbit and Sierra, the write throughput becomes larger as there are more nodes to serve writing the primary data.

5. Conclusion and Future Work

Recently, energy-efficient infrastructures for Big Data processing are gaining much attention from both industrial and academia. In this paper, we identified the issue of ineffective gear-shifting in power-proportional distributed file systems and we proposed the Accordion data placement method to address this problem. Furthermore, the Accordion configuration is highly flexible because the number of nodes in each group can be determined by a weak constrain. In Accordion, the data reliability in the lowest configuration which is omitted in other methods, is ensured through utilizing chained declustering. Extensive experiments using actual machines with the Accordion prototype verified the effectiveness of Accordion. Accordion also reduced the execution time required for updated data movement by 66% and improved the power-proportional performance by 30% compared with Rabbit and Sierra, two of the most standard power-proportional data placement aiming for Big Data. Furthermore, Accordion is applicable for smooth gear-shifting in multigear systems, especially when the number of gear is high. In such situation, when the gear-shifting is occurred at a high gear, Accordion was shown to be able to effectively shorten the execution time by at most 91% compared to Sierra. From the experiment results, Accordion is believed to be capable for deploying in real system, especially for commodity-off-the-shelf based systems, in which the power consumption is still not gained enough support from modern hardware technologies.

In the future, we would like to confirm the effectiveness of Accordion in different experimental environments, by using actual benchmarks and with more number of nodes. Moreover, we want to integrate Accordion with architectures other than HDFS. We will also consider developing a specific algorithm to deal with system failures.

Acknowledgments

This work was supported partly by Grants-in-Aid for Scientific Research from the Japan Science and Technology Agency (A) (#22240005).

References

[1] H.H. Le, S. Hikida, and H. Yokota, "Efficient gear-shifting for a power-proportional distributed data-placement method," Proc. 2013 IEEE International Conference on Big Data, pp.76–84, 2013.

[2] S. Ghemawat, H. Gobioff, and S.T. Leung, "The google file system," Proc. 19th ACM Symposium on Operating Systems Principles, pp.29–43, 2003.

[3] Apache Hadoop, "HDFS Hadoop Wiki." http://wiki.apache.org/hadoop/HDFS.

[4] L.A. Barroso and U. Hölzle, "The case for energy-proportional computing," Computer, vol.40, pp.33–37, 2007.

[5] W. Charles, O. Mathew, Q. Jin, W.A.I. Andy, R. Peter, and K. Geoff, "PARAID: A gear-shifting power-aware RAID," ACM Trans. Storage, vol.3, pp.13:1–13:33, 2007.

[6] A. Hrishikesh, C. James, G. Varun, G.R. Ganger, K. Michael A., and S. Karsten, "Robust and flexible power-proportional storage," Proc. 1st Symposium on Cloud Computing, pp.217–228, 2010.

[7] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: Practical power-proportionality for data center storage," Proc. 6th European Conference on Computer Systems, pp.169–182, 2011.

[8] S. Priya, T. Vasily, and Z. Erez, "Evaluating performance and energy in file system server workloads," Proc. 8th USENIX Conference on File and Storage Technology, pp.253–266, 2010.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Proc. 6th Symposium on Operating System Design and Implementation, pp.137–150, 2004.

[10] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," ACM Transaction on Storage, vol.4, no.3, pp.10:1–10:23, 2008.

[11] H.H. Le, S. Hikida, and H. Yokota, "An evaluation of power-proportional data placement for Hadoop distributed file systems," Proc. Cloud and Green Computing, pp.752–759, 2011.

[12] H.H. Le, S. Hikida, and H. Yokota, "NameNode and DataNode coupling for power-proportional Hadoop distributed file system," Proc. 18th International Conference on Database System for Advanced Applications, Part II, pp.99–107, 2013.

[13] H.H. Le, S. Hikida, and H. Yokota, "NDCouplingHDFS: A coupling architecture for a power-proportional Hadoop distributed file system," IEICE Trans. Inf. & Syst., vol.E97-D, no.2, pp.213–222, Feb. 2014.

[14] B. Mao, D. Feng, H. Jiang, S. Wu, J. Chen, and L. Zeng, "GRAID: A green RAID storage architecture with improved energy efficiency and reliability," Proc. 16th International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, pp.1–8, 2008.

[15] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of Hadoop clusters," SIGOPS Operating System Review, vol.44, pp.61–65, 2010.

[16] R.T. Kaushik and B. Milind, "GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster," Proc. 2010 International Conference on Power Aware Computing and Systems, pp.1–9, 2010.

[17] J. Kim and D. Rotem, "Energy proportionality for disk storage using replication," Proc. 14th International Conference on Extending Database Technology, pp.81–92, 2011.

[18] H.I. Hsiao and D. DeWitt, "Chained declustering: A new availability strategy for multiprocessor database machines," Proc. 6th International Conference on Data Engineering, pp.456–465, 1990.

[19] J. Gray, "Greetings from a filesystem user," Proc. 4th USENIX Conference on Files and Storage Techniques, 2005.

[20] "AC/DC Power HiTester 3334." http://www.hioki.com/products/power_current_sensor/power_meters/402.

Hieu Hanh Le received his B.E., and M.E. degree from Tokyo Institute of Technology in 2008, and 2010, respectively. He is currently a researcher at Yokohama Research Laboratory of Hitachi. He is interested in research on data engineering, information storage systems, and network engineering. He is a member of IPSJ.

Satoshi Hikida received his M.E. degree from Tokyo Institute of Technology in 2011. He is currently a Ph.D student at Tokyo Institute of Technology. He is engaged in research on data engineering, and information storage systems. He is a student member of IPSJ.

Haruo Yokota received his B.E., M.E. and Dr.Eng. degrees from Tokyo Institute of Technology in 1980, 1982, and 1991, respectively. He joined Fujitsu Ltd. in 1982, and was a researcher at ICOT for the Japanese 5th Generation Computer Project from 1982 to 1986, and at Fujitsu Laboratories Ltd. from 1986 to 1992. From 1992 to 1998, he was an Associate Professor at Japan Advanced Institute of Science and Technology (JAIST). He is currently a Professor at the Department of Computer Science in Tokyo Institute of Technology. His research interests include the general research areas of data engineering, information storage systems, and the dependable computing. He was a chair of ACM SIGMOD Japan Chapter, a trustee board member of IPSJ and the DBSJ, and the Editor-in-Chief of Journal of Information Processing. He is a Vice Chair of DBSJ, an associate editor of the VLDB Journal, a fellow of IEICE and IPSJ, senior member of IEEE, and a member of JSAI, ACM, and ACM-SIGMOD.